Skrajšani zapiski

Da se tisti, ki že znate programirati, ne boste prebijali čez zapiske, katerih dolžine se ne bi sramovala ne Fjodor Mihajlovič ne Lev Nikolajevič, je tu bistveno skrajšana verzija, v kateri skoraj nič na manjka.

Python v konzoli

Na prvih predavanjih običajno poženem `python` (npr. iz konzole/lupine/terminala/ukaznega pozivnika) in potem direktno vpisujem stvari vanj. Vpišeš `1 + 1` (Enter) in odgovori 2. Vpišeš `x = 3 + 4` in ne odgovori ničesar. Napišeš `x` (Enter) in reče 7. Napišeš `abs(x - 10)` in reče 3 (če je `x` enak 7).

To je luštno zato, ker lahko sproti preskušamo kakšne izraze, funkcije ipd. Za predavanja pa je sploh praktično.

Števila

Python ima cela in necela števila, ampak samo po ena vsake vrste. Tipa se imenujeta `int` in `float`. `int` je lahko poljubno velik in ne le 2^{32} ali 2^{64}, kot je to običajno v drugih jezikih. Lahko izpišete `2 ** 1000000`, če čte (`**` pomeni potenciranje). `float` ima osem bajtov; v kakem drugem jeziku bi to prodajali že kot `double`.

`+`, `-`, `*` in `%` so takšni, kot ste jih navajeni. `**` je potenciranje.

Rezultat `/` je vedno `float`, tudi, če delimo `int`-e. `8 / 3` je `2.6666666665`, ne 2. Pač pa ima Python še `//` za celoštevilsko deljenje. `8 // 3` je `2` in `4.5 // 1.2` je `3.0` (`float`, ampak celo število). Predvsem obnašanje `/` je kar fino, ker se v drugih jezikih zmotiš tako, da ponesreči deliš `int`-e in ti potem ni jasno, zakaj ne dela. Obratno se zgodi redkeje.

Nizi

Python ima samo nize (tip se imenuje `str`), ne pa posameznih znakov (a la `char`).

Enojni in dvojni narekovaji so si enakovredni. Vseeno je, katere uporabljamo, samo da niz, ki ga začnemo z enojnimi tudi končamo z enojnimi in obratno. To je dobro zaradi:

```
sentence = "Don't do this."
levstik = 'Cesar reče: "Pa srečno hodi," minister pa nič.'
```

Nize seštevamo s `+`. Lahko jih tudi množimo s številom. `"Ana" * 3` vrne `"AnaAnaAna"`, kar je logično, saj je `"Ana" * 3` pač isto kot `"Ana" + "Ana" + "Ana"`. To je uporabno, da z `"-" * 10` dobimo `"----------"`.

Če hočemo pretvoriti niz v število, pokličemo `int` ali `float`. `int("13")` vrne `13` in `float("3.14")` vrne `3.14`. Obratno poskusite sami. Aha, tudi `int(3.67)` je

`3`. (Ozadje: malo podobno kot v C++ ali Kotlinu (in malo manj kot v Javi ali JavaScriptu) so tudi v Pythonu imena tipov pravzaprav konstruktorji. Samo da je v Pythonu tu zadaj še malo globja filozofija.)

V drugo smer: `str(13)` vrne `"13"`.

Python ni JavaScript. `"1" + 1` ne vrne niza `"11"` temveč javi napako.

Spremenljivke

Pomen spremenljivk je malo drugačen kot v Javi ali C/C++, ter bolj podoben spremenljivkam v JavaScriptu. In običajno ne rečemo "spremenljivka", temveč "ime". Zakaj, povem čez par tednov.

Spremenljivk ni potrebno vnaprej deklarirati. Preprosto napišemo `x = 5` ali `x = 3.14`. V prvem primeru se ime `x` nanaša na nek `int`, v drugem na nek `float`. Isto ime se lahko v istem programu/funkciji nanaša najprej na en tip, potem na drugega. Tole je legalen program.

```
x = 5
x = 3.14
x = "Ana"
print(x)
```

```
Ana
```

Kot sem napisal: `x` ni spremenljivka v C-jevskem/Javanskem pomenu. (Python je *strogo dinamično tipiziran jezik*. *Dinamično*, ker se isto ime lahko nanaša na različne tipe, *strogo*, ker noče sešteti `"1" + 1`. Za primerjavo: Javascript = šibko dinamično, C#/C++/Java = strogo statično, Kotlin = nacistično statično, Php = konfuzno.)

Imena spremenljivk so, kot običajno, sestavljena iz črk, podčrtajev in števk (samo ne na začetku). Črke so lahko tudi slovenske, ruske ali japonske, ampak tega ne počnite.

V Pythonu je običaj, da spremenljivke pišemo z malimi črkami in med besede dajemo podčrtaj. Se pravi `visina_triglava` in ne `visinaTriglava` (kot bi bilo običajno v Cju, Javi, Javascriptu) ali celo `VisinaTriglava` (yay, basic!). Tak je pač dogovor. Navadite se, da v vsakem jeziku programirate tako, kot je v tem jeziku navada. Te navade navadno niso same sebi namen.

Izogibajte se imenom O in l, ker jih je v nekaterih pisavah težko ločiti od 0 in 1.

Izogibajte se imenom, kot so `a`, `aa`, `bbb`... ki ne povedo nič. Če bom na predavanjih kdaj počel kaj takega, je to samo zato, da ne bom predolgo tipkal.

Ime naj pove vsebino, ne vrste. Ko bomo začeli uporabljati sezname, bodo mnogi imeli spremenljivko `seznam`, in, če bo seznamov več, se bodo imenovali `seznam1`, `seznam2`, `seznam3`. A ne bi bilo boljše, če bi ime povedalo *kaj* je v katerem od njih, in ne le, da gre za neke sezname?

Funkcije

Funkcije pokličemo kot v vseh normalnih jezikih, z oklepaji: `abs(-3.14)`.

Python ima nekaj vdelanih funkcij, npr. `abs`, `len`, `print`, `input`... Teh ni potrebno uvažati. Nobenega C-jevskega `#include <stdio.h>`.

Vse drugo je, kot ste najbrž že navajeni, po raznih modulih. Za začetek jih bomo na hitro uvažamo z ukazi, kot npr. `from math import *`, da dobimo `sin`, `cos`, `log` ..., čez nekaj tednov pa se bomo naučili pravilnejšega uvažanja.

Vpis in izpis

Funkcija `print` prejme poljubno število argumentov poljubnih tipov in izpiše njihove vrednosti. Vmes doda presledke. To izgleda sitno, ampak je praktično, saj bomo takrat, ko jih ne bomo hoteli, itak naredili nekaj drugega.

`print(6, "krat", 8, "je", 42)` izpiše `"6 krat 8 je 42"`. Čeprav ni.

Funkcija `input` prejme niz, počaka, da uporabnik kaj tvipka in vrne tipkano. Kot niz, četudi je uporabnik morda vpisal šteivlo.

Program

```
temp_C = input("Temperatura [C]? ")
temp_F = float(temp_C) * 5 / 9 + 32
print(temp_C, "C je enako", temp_F, "F")
```

```
Temperatura [C]? 15
15 C je enako 40.333333333333336 F
```

Na konce vrstic ne pišemo podpičij. Lahko jih, ampak vas bodo čudno gledali.

Na začetke vrstic ne smemo dajati presledkov. Zamikati moramo tam (in smemo samo tam), kjer je treba.

Program lahko natipkamo v poljubnem Notepadu, Textpadu, VS Codeu, Sublimeu, vi-ju, emacsu ... in ga poženemo s `python ime_datoteke_s_programom`. Raje pa ga vtipkamo v PyCharm in poženemo s Ctrl-Shift-F10 (Cmd-Shift-R na macOS).

Programi v Pythonu se ne prevajajo. Točneje, se, ampak tik pred izvajanjem, zato izgleda, kot da se ne. Zato python ni prevajalnik (compiler) temveč "tolmač" (interpreter).